CRAY XMS™ Systems IOS™ Reference Manual

SR–3085 1.1

Cray Research, Inc.

Requests for copies of Cray Research, Inc. publications should be sent to the following address:

Cray Research, Inc.
Distribution Center
2360 Pilot Knob Road
Mendota Heights, MN 55120

Order desk (612) 681–5907
Fax number (612) 681–5920

Record of Revision

The date of printing or software version number is indicated in the footer. In reprints with revision, changes are noted by revision bars along the margin of the page.

Version	***Description***
1.1	March 1991 – Original printing.

Preface

This publication is for system administrators or operators of the CRAY XMS system. It is to be used after UNICOS 5.1 and the IOS have been installed on the CRAY XMS system.

Conventions in this manual

This manual uses the following typographical conventions:

Convention	Description
command(1M)	The designation (1M) following a command name indicates that the command is documented in *UNICOS Administrator Commands Reference Manual*, publication SR–2022.
italics	Italics indicate variable or user-supplied information or a term being defined.
`typewriter`	This typewriter-like font indicates literal input to and output from a computer.
`bold typewriter`	Bold indicates what you type in an interactive session.
[ ]	Brackets enclose optional portions in syntax lines.

Other syntax conventions pertaining to the IOS on the CRAY XMS system are described in detail in Section 2, “Commands Summary”.

Related publications

The CRAY XMS system runs UNICOS 5.1 operating system similar to any other Cray Research systems. Therefore, you will find the entire UNICOS manual set to be helpful. However, because there are some differences that are especially important to the system administrator and operator, the following CRI publications help you successfully run UNICOS on the CRAY XMS system:

- *CRAY XMS Computer System On-Line Diagnostics Maintenance Manual*, publication SMM–1023 [§]
- *Differences for UNICOS 5.1 on the CRAY XMS Systems*, publication SN–3086
- *A Brief Overview of Your CRAY XMS System*, publication SG–3093
- *UNICOS 5.1 System Installation Bulletin for CRAY XMS Systems,* UC–05.1–UDN–SIB

Reader comments

If you have comments about the technical accuracy, content, or organization of this manual, please tell us. You can contact us in any of the following ways:

- Call our Software Information Services department at (612) 683–5729.
- Send us electronic mail from a UNICOS or UNIX system, using the following UUCP address:

```
uunet!cray!publications
```

§ CRAY RESEARCH PROPRIETARY. Dissemination of this document to non-CRI personnel requires approval of the appropriate vice president and that the recipient sign a nondisclosure agreement. Export of technical information in this category may require an export license.

- Send us electronic mail from any system connected to Internet, using the following Internet addresses:

  `pubs3085@timbuk.cray.com` (comments specific to this manual)

  `publications@timbuk.cray.com` (general comments)

- Send a facsimile of your comments to the attention of "Software Information Services" at fax number (612) 683–5599.

- Use the postage-paid Reader's Comment form at the back of this manual.

- Write to us at the following address:

  Cray Research, Inc.
  Software Information Services Department
  655F Lone Oak Drive
  Eagan, MN 55121

We value your comments and will respond to them promptly.

Contents

Introduction [1]

Page

Introduction [1]

This manual describes the I/O Subsystem (IOS) on the CRAY XMS system, which is significantly different from the IOS on any other Cray Research system running UNICOS. More specifically, the major differences are as follows:

- The CRAY XMS system has only one I/O processor. There is no MIOP, BIOP, DIOP, or XIOP. All types of peripherals use one IOP.
- There is only one terminal connecting to the IOS port. This terminal serves as the IOS console and the UNICOS system console.
- The IOS does not log information or keep statistics about channel use, error detection, and recovery in the same manner or scope.
- The IOS runs completely different underlying software.

Capabilities of the IOS on the CRAY XMS system
1.1

From the IOS console, you have the following capabilities:

- Control of the CPU state
- Read and write access to CPU central memory
- Deadstart capability
- Access to all peripherals
- CPU debugging tools and diagnostics

Control of the CPU state
1.1.1

You have the ability to run, halt, or deadstart the CPU. The operator console is used to communicate with UNICOS 5.1 after startup procedures have completed.

Read and write access to CPU central memory
1.1.2

You have the ability to view and/or change memory location values on the CPU through the IOS console.

Deadstart capability
1.1.3

You can start the CPU from an initial execution through a hardware startup procedure. The CPU, once it has deadstarted, will begin execution in order to bootstrap and load the operating system.

Access to peripherals
1.1.4

You can access peripheral devices for diagnostic, maintenance, and operational purposes.

CPU debugging tools and diagnostics
1.1.5

You can run both off-line diagnostic and debugging tools from the IOS console.

Commands Summary [2]

Commands Summary [2]

This section describes console commands that control the I/O Subsystem (IOS) on the CRAY XMS system. More specifically, it contains the following:

- Command syntax
- File specifications
- Characteristics of console commands
- Noninteractive commands
- Commands summary grouped by function

This section is a summary only. For complete information on each command, see Appendix A, "Man Pages," page 13, which contains the printed version of the on-line man pages.

Command syntax
2.1

The command syntax is as follows:

```
command parameter1,parameter2,parameter3,parameter4
```

You can enter commands and parameters in uppercase, lowercase, or mixed-case. Separate commands and parameters with either a comma or a space. If default values are to be used for any parameter, an empty field for that parameter should be included by using two successive commas (for noninteractive commands).

File specifications
2.2

The I/O Subsystem supports a hierarchical file system. A file specification (*filespec*) consists of three parts:

1. Path name
2. File name
3. Extension

The *pathname* identifies the path in a tree structure of a hierarchical file structure. The elements in the path name consist of a series of subdirectories located between the root and the target directories. The subdirectories are separated with a slash (/). Path names can be relative or absolute.

A *filename* consists of 1 to 8 characters, and it is not case sensitive (for example, `BAS2ST.CMP` and `bas2st.cmp` refer to the same file). Enter the file name in uppercase or lowercase. ASCII characters that are less than 20 (hexadecimal) or any of the following characters are invalid in file names:

```
. " / \ [ ] | < > + = ; ,
```

The special characters `?` and `*` are permitted in file names or extensions as metacharacters. The `?` in a file name indicates that any character can occupy that position. An `*` in a file name or extension indicates that any character can occupy that position and all the remaining positions in the file name or extension.

An *extension* consists of a period followed by 1 to 3 valid characters.

Characteristics of console commands
2.3

All IOS console commands share the following characteristics:

- The command prompt is `RT>`.
- After a command executes, the prompt reappears on the screen. If no error messages appear before the prompt reappears, the command has been executed successfully.
- Commands must be executed by a carriage return (the RETURN on the terminal).

Noninteractive commands
2.4

Noninteractive commands do not prompt for their parameters and have the following characteristics:

- Commands are usually followed by one or more parameters.
- Commands and parameters must be separated by delimiters. A delimiter is a comma or one or more spaces. The delimiters can be mixed within one command. For example, the following command line is acceptable:

  ```
  AMS 1000,1111  2222,3333 4444
  ```

- Parameters are order dependent. If an optional parameter is the last parameter in a command, it may be omitted. If it is not the last, and it will not be specified, enter two commas in a row, and the default for that parameter will be used. For example, to display 5 memory words starting at the default word address, use the following:

  ```
  DM, ,5
  ```

Commands summary grouped by function
2.5

The commands in each subsection are described in alphabetical order in each of the following categories:

- Help command
- Shell built-in commands
- Breakpoint commands
- CPU state commands
- Disk commands
- File utility commands
- Memory commands
- Performance monitor commands
- Miscellaneous commands

For a complete command description, see Appendix A, "Man Pages," page 13, which consists of an alphabetical listing of the printed man pages for these commands. To view these man pages on-line, type `man` *command* on a terminal that has access to UNICOS.

Caution

Commands that are preceded by an asterisk (`*`) may cause nonrecoverable errors. Do not use these commands unless you have a full understanding of their effects.

Help command 2.5.1

The following command helps you with correct console command syntax.

Command	Description
`HELP` or `?`	Lists all console commands and their syntaxes

Shell built-in commands 2.5.2

The following shell commands can be used in a shell script. The I/O Subsystem for the CRAY XMS has a built-in interpreter that reads each file. If the file contains a series of commands, these commands will be executed. You can create a command file by using the `ed` editor, and execute the file by entering the file name at the IOS prompt. You can also use `exec -x` *filename* to set a debug flag in your script.

Command	Description
`CONSWITCH`	Toggles console from IOS to UNICOS
`COUNT`	Counts the number of passes executed by a loop
`ECHO`	Displays a message
`EXEC`	Executes a shell script file
`GOTO`	Transfers control to the line following the one containing the appropriate label
`IF`	Allows conditional transfer of control
`TEST`	Returns the program counter `P` or state of the `PMATCHED` flag
`WAIT`	Waits *n* seconds, then returns

Breakpoint commands 2.5.3

The following commands help you manipulate breakpoint.

Command	Description
BPC	Clears breakpoint
BPD	Disables breakpoint
*BPE	Enables breakpoint
BPL	Lists breakpoint
*BPS	Sets breakpoint

CPU state commands 2.5.4

The following commands allow you to control the state of the CPU.

Command	Description
AP	Alters scan page
AF	Alters clock cycle frequency
*AR	Displays and changes registers and buffers
DIOQ	Displays I/O request queue
DLBA	Displays last branch addresses
DREG	Dumps CPU registers to IOS screen
*RUPT	Interrupts the CRAY XMS CPU from the console
*SER	Toggles serial mode on or off
STAT	Displays CPU states every 0.5 seconds
STEP	Executes instructions one at a time

Disk commands 2.5.5

The following commands allow you to manipulate your disks.

Command	Description
DADISABLE	Disables DA drive and substitutes hot standby drive
DAFLAWR	Reads the Disk Array Flaw table
DAFLAWW	Writes the Disk Array Flaw table
DAFORMAT	Formats the disk array
DAVERIFY	Scans drive to verify media
DAREPLACE	Reconstructs data on a newly replaced disk

Command	Description
DAREPLACE	Reconstructs data on a newly replaced disk
DAWCONFIG	Writes disk array configuration to controller
DDFLAWR	Reads flaw map and writes it to a file
DDFORMAT	Formats the ESDI hard disk
DDVERIFY	Detects and marks bad blocks

File utility commands 2.5.6

The following commands allow you to manipulate your files.

Command	Description
CAT	Lists text file to screen
CD	Changes current directory
CP	Copies a file
ED	Edits a text file
FSF	Spaces forward one file on a tape
HEAD	Displays first few lines of a file
LDF	Transfers file from tape to disk
LS	Lists a directory
MKDIR	Creates a subdirectory
MKFS	Formats a hard disk
MORE	Displays a file one screen at a time
MOUNT	Mounts local Winchester drive
MV	Moves (renames) a file
PWD	Prints current directory
RENAME	Changes a file name
RM	Removes a file
RMDIR	Removes a subdirectory
TAR	Archives tape files
UMOUNT	Unmounts local Winchester drive

Memory commands
2.5.7

The following commands allow you to control central memory. Many of these commands can access memory two ways: by using the I/O channel or by using the scan path. The commands ending in `S` use the scan path. For example, `AM` and `AMS` both alter central memory. However, `AM` uses the I/O channel, and `AMS` uses the scan path.

Command	Description
`*AM`	Alters central memory by using I/O channel
`*AMS`	Alters central memory by using scan path
`*AMVE`	Alters IOS memory
`BASE`	Changes or displays the base address
`CMT`	Compares central memory with the contents of a text file by using I/O channel
`CMTS`	Compares central memory with the contents of a text file by using scan path
`DM`	Displays central memory by using I/O channel
`*DMS`	Displays central memory by using scan path
`DUMP`	Captures mainframe memory
`*DXP`	Displays exchange packet by using I/O channel
`*DXPS`	Displays exchange packet by using scan mode
`*FM`	Fills central memory by using I/O channel
`*FMS`	Fills central memory by using scan path
`IOSDUMP`	Dumps memory to file on SCSI disk
`*LM`	Loads central memory from ESDI disk by using I/O channel
`*LMT`	Loads central memory with the contents of a text file by using the I/O channel
`*LMTS`	Loads central memory with the contents of a text file by using scan path
`MM`	Matches central memory
`MODE`	Changes the radices for the `DM` command
`*RST`	Turns the reset on or off (if file name is specified, loads central memory with contents of the scan path)
`*RSTS`	Turns the reset on or off (if file name is specified, loads central memory with contents of the file by using file using the I/O path)

Command	Description
RUN	Loads a program from a text file to central memory; runs, then stops and compares central memory with another text file by using I/O channel
RUNS	Loads a program from a text file to central memory; runs, then stops and compares central memory with another text file by using scan path
SM	Saves central memory to ESDI disk by using I/O channel
SMT	Saves central memory contents to a text file by using I/O channel
*SMTS	Saves central memory contents to a text file by using scan path

Performance monitor commands 2.5.8

The following commands allow you to manipulate the performance monitor counters.

Command	Description
PML	Lists performance monitor counters
PMS	Sets performance monitor counters

Miscellaneous commands 2.5.9

The following commands are used to start and stop the IOS, and communicate with the operating system.

Command	Description
AC	Alters scan chain
*CLK	Turns the clock on or off, or steps the clock
DEBUG	Sets a debug flag in the IOS
DR	Displays the P or PMATCHED
*IOSINIT	Initializes the CRAY XMS system
IOSTART	Starts communication between IOS and UNICOS
IOSTOP	Stops communication between IOS and UNICOS
IU	Turns instruction unit on or off

Command	Description
`LOAD`	Loads and boots IOS binary image
`RELOAD`	Initiates reboot
`REWIND`	Rewinds tape
`STD`	Reads time and date from IOS and writes to central memory
`SYNC`	Flushes outstanding I/O to hard disk
`TIME`	Sets and displays the real-time clock in the I/O subsystem
`VER`	Displays version number of the IOS

Man Pages [A]

Page

13 Text file format

Man Pages [A]

This section contains hardcopy versions of the on-line man pages for the IOS console commands ordered alphabetically. See Section 2, "Commands Summary," page 3, for a listing of the commands in terms of functionality. For information regarding command conventions, see "Command syntax," page 3. To view man pages on-line, type `man` *command* on a system running UNICOS.

Text file format
A.1

Some commands expect a text file format. The format for `.XXX` and `.CMP` files follows:

A text file which is prepared for loading into central memory will accept the two possible line types as follows:

- A line specifying the address at which the load is to start
- A line containing data

An address line starts with an asterisk and follows with the address in hexadecimal, as in the following example:

```
*12AB
```

A data line contains four hexadecimal "parcels" labeled `A`, `B`, `C`, and `D` with `A` being the leftmost parcel and `D` the rightmost parcel, as in the following example:

```
0112 12AB A1EC DC34
```

In the preceding example, the most significant number is a 0 at the left end, and the least significant number is a 4 at the right end. The 4 parcels comprise a 64-bit word of central memory.

A file prepared for loading into memory can have several address lines and several data lines. Each address line must specify the starting address for loading the data in the data lines which follow. Each occurence of an address line will specify a new start-load address.

A file that results from a memory dump can have only one address line. The syntax for a memory read permits only one address to be specified.

NAME

ac – Alters chain scan

SYNOPSIS

ac *chain*

DESCRIPTION

The **ac** command examines and modifies a chain scan.

chain	Name of chain to be brought up. (Default: IU for the first time, or the last scan chain displayed.) Valid chain names are: IU, MC1, MC2, V0, V1, V2, V3, VC, ADR, SC, CSI, IO1, IO2, FR1, FR1.

NOTES

This command accesses central memory by scanning; therefore, the clock must be off.

A chain displayed on the screen is ready to be modified as needed. The top line lists the functions that can be performed and the respective keys. The name of the chain is displayed on the next line.

If SYSRST signal is ON, a warning message is displayed.

Each 1 or 0 represents a flip-flop in the scan chain. A copy of each chain is kept in the IOS memory. When a board is first displayed, it reflects the state of the hardware in the CRAY XMS system. Each individual flip-flop can be set or reset by modifying it on the screen. This only changes the copy in the IOS memory. To actually alter the chain, you must scan in the chain (F5). To ignore all changes since the last scan in or scan out, press F7 to scan out.

To alter a flip-flop, enter a 1 or a 0 from the keyboard at the cursor position. The arrow keys up, down, left, and right can be used to move around in the chain.

To ensure that the copy in the IOS memory and the hardware are the same, when you leave one chain to bring up another (F1, F3, or F4), or exit **ac** command (F10), the chain is checked. If the chain has not been scanned in or out since it has been changed, a message is displayed on the screen. You must go back and either scan in or out before executing those functions. Continuously scanning in (F6), continuously scanning out (F8), and checking (F9) functions also synchronize the chain buffer and the hardware.

The following is a description of what each function key does:

F1	Brings up a different chain
F2	Fills the chain with an 8-bit pattern
F3	Brings up the previous chain
F4	Brings up the next chain
F5	Scans in the chain currently on the screen
F6	Continuously scans in the chain until a key is pressed
F7	Scans out the chain and updates the screen
F8	Continuously scans out the chain until a key is pressed

F9 Automatically checks the chain currently on the screen. There are 4 patterns that the IOS checks: 0x00, 0x55, 0x5a, 0xA5. The current chain is filled with 0x00, scanned in, then scanned back out. The data scanned out is compared with the data scanned in. If no mismatch is found, the next pattern is used, until all 4 patterns are checked out. Otherwise, there must be scan hardware problems, and an error message is displayed.

F10 Exits the **ac** command

NAME

am – Alters memory

SYNOPSIS

am *address*, [*parcelA*], [*parcelB*], [*parcelC*], [*parcelD*]

DESCRIPTION

The **am** command alters the contents of a 64-bit word in central memory.

address	Relative memory address to be altered.
parcelA	Value of parcel to alter memory (most significant). (Default: no change.)
parcelB	Value of parcel to alter memory. (Default: no change.)
parcelC	Value of parcel to alter memory. (Default: no change.)
parcelD	Value of parcel to alter memory (least significant). (Default: no change.)

EXAMPLES

This command writes the value 1111 2222 3333 4444 to central memory word 1000 hexadecimal.

```
am 1000,1111,2222,3333,4444
```

NOTES

This command accesses central memory through the data channels; therefore, the CPU clock must be ON.

NAME

ams – Alters memory using scan

SYNOPSIS

ams *address*, [*parcelA*], [*parcelB*], [*parcelC*], [*parcelD*]

DESCRIPTION

This command alters the contents of a 64-bit word of central memory by using the scan path. All parameters are entered in hexadecimal.

address	Relative memory address to be altered.
parcelA	Value of parcel to alter memory (most significant). (Default: no change.)
parcelB	Value of parcel to alter memory. (Default: no change.)
parcelC	Value of parcel to alter memory. (Default: no change.)
parcelD	Value of parcel to alter memory (least significant). (Default: no change.)

EXAMPLES

The following command line writes the value 1111 2222 3333 4444 to central memory word 1000 hexadecimal.

```
ams 1000,1111,2222,3333,4444
```

NOTES

This command accesses central memory by scanning; therefore, the CPU clock must be OFF.

NAME

amvme – Alters IOS memory

SYNOPSIS

amvme *add data*

DESCRIPTION

The **amvme** command alters the IOS memory.

add	IOS memory address
data	Data to write to IOS memory

NAME

ap – Alters scan page

SYNOPSIS

ap [*page1*] [*page2*]

DESCRIPTION

The **ap** command provides a way to examine and modify the logical pages.

page1	Name of the logical page to be displayed on the upper half of the screen. (Default: IU1 for the first time, or the last scan page displayed otherwise.)
page2	Name of the logical page to be displayed on the lower half of the screen. (Default: IU2 for the first time, or the last scan page displayed otherwise.)

Valid page names are: I1, I2, I3, M1, M2, M3, V1, V2, V3, S1, SA, A1, CS, NC, FR.

NOTES

The screen is divided into two halves, top and bottom. A scan page is displayed in each half. A page displayed on the screen is ready to be modified as needed. The bottom line lists the functions that can be performed and the respective keys. Each page has a maximum of 44 entries, with 1, 2, 3, or 4 entries per line. Each entry is divided into two sides; the left side is the logical name of the signals; the right side lists the values of the signals. Each digit in the value column is a hexadecimal number, representing up to four signals.

Before a page is displayed, all boards are scanned out. When the **ap** command is exited with the F10 key, all boards are scanned in before any new command is accepted. Each individual signal can be set or reset by modifying it on the screen.

When a page is modified, only the copy in the IOS memory is modified. The function clock once (F1), or the function clock *n* times (F2) scans in all the boards in the the CRAY XMS system before issuing the clock command. To discard all changes, use the function reread (F5).

The arrow keys up, down, left, and right can be used to move around on the page. Following is a description of what each function key does:

F1	Clocks once
F2	Clocks *n* times
F3	Brings up the previous page
F4	Brings up the next page
F5	Scans out all the boards, and updates the page on the screen
F7	Swaps the upper screen to another page
F8	Resets the count of the number of clock cycles issued
F9	Swaps the lower screen to another page
F10	Exits the **ap** command

NAME

ar – Alters register

SYNOPSIS

ar [*regname*]

DESCRIPTION

The **ar** command provides a way to examine and/or modify register values.

regname Name of register to be examined and/or modified. (Default: **A**, **B**, and **S** registers for the first time, or the last register(s) displayed.) Valid register names are: **a**, **s**, **b**, **t**, **v0-7**, **sm**, **sb**, **st**, **io-3**.

EXAMPLES

```
F1=Save  F2= Reg  F3=Prev  F4=Next  F10=Exit

A0= 5D5D32          S0= 5D5D5D5D 5D5D5D5D

A1= 325D5D          S1= 5D5D5D5D 5D5D325D

A2= 5D5D5D          S2= 5D325D5D 5D325D5D

A3= 5D5D5D          S3= 5D5D5D5D 325D5D5D

A4= 5D5D32          S4= 5D325D5D 5D5D325D

A5= 5D5D5D          S5= 325D5D5D 5D325D5D

A6= 5D5D32          S6= 5D5D5D5D 5D5D5D32

A7= 5D5D5D          S7= 5D325D5D 5D5D5D5D

B00= D5D325 D5D3D5 D5D5D5 D325D5 D5D5D5 D5D5D3 D5D5D5D

B10= D5D325 D5D3D5 D5D5D5 D325D5 D5D5D5 D5D5D3 D5D5D5D

B20= D5D325 D5D3D5 D5D5D5 D325D5 D5D5D5 D5D5D3 D5D5D5D

B30= D5D325 D5D3D5 D5D5D5 D325D5 D5D5D5 D5D5D3 D5D5D5D

B40= D5D325 D5D3D5 D5D5D5 D325D5 D5D5D5 D5D5D3 D5D5D5D

B50= D5D325 D5D3D5 D5D5D5 D325D5 D5D5D5 D5D5D3 D5D5D5D

B60= D5D325 D5D3D5 D5D5D5 D325D5 D5D5D5 D5D5D3 D5D5D5D

B70= D5D325 D5D3D5 D5D5D5 D325D5 D5D5D5 D5D5D3 D5D5D5D
```

NOTES

This command is at the scan level; therefore the CPU clock must be OFF.

All registers are displayed in hexadecimal, except **I0-3** which is displayed in octal. All registers can be changed by moving the cursor to the desired register and entering the desired value at the cursor position, then saving the values by using the F1 function key.

The following list describes the operation of the function keys:

F1	Saves
F2	Brings up another set of registers
F3	Brings up the next set of registers
F4	Brings up the previous set of registers
F10	Exits the **ar** command

NAME

base – Sets or displays base addresses

SYNOPSIS

base [*address*]

DESCRIPTION

The **base** command sets or displays the base address for the commands **am**, **ams**, **fm**, **fms**, **dm**, and **dms**. The base address is added to the address parameter of those commands to obtain the physical address. If no parameter is entered, the current base address is reported.

address	Base address.

EXAMPLES

The following command line sets the base to 4321 hexadecimal.

```
base 4321
```

NAME

bpc – Clears breakpoint enables

SYNOPSIS

bpc p|or|ir|ow|iw

DESCRIPTION

The **bpc** command clears breakpoint enables. Breakpoint addresses are unchanged.

p	Clears P address compare enable.
or	Clears operand read address compare enable.
ir	Clears I/O read address compare enable.
ow	Clears operand write address compare enable.
iw	Clears I/O write address compare enable.

EXAMPLES

The following command line disables the program counter address compare:

```
bpc p
```

NOTES

IU must be off or RST must be on.

NAME

bpd – Disables all actions for breakpoint matched

SYNOPSIS

bpd

DESCRIPTION

The **bpd** command disables all actions for breakpoint matched (that is, turns off both I and C).

NOTES

IU must be off or RST must be on.

NAME

bpe – Enables action for breakpoint matched

SYNOPSIS

bpe i|c|ic

DESCRIPTION

The **bpe** command takes the specified action upon a breakpoint match.

i	Stops instruction issue on any address match.
c	Stops the clock on any instruction match.
ic	Both of the above.

EXAMPLES

This command enables a break when an instruction match occurs. At the break, the clock will be stopped. The address will have been set by using the **bps** command.

```
bpe c
```

NOTES

IU must be off or RST must be on.

NAME

bpl – Lists breakpoint information

SYNOPSIS

bpl

DESCRIPTION

The **bpl** command displays the current breakpoint information.

EXAMPLES

```
   PADR1 = 000000    PADR2 = 000000  ENP        = 0
OPNDADRL = 0000000 PNDADRH = 000000  ENOPNDR    = 1  ENOPNDW= 0
  IOADRL = 000000   IOADRH = 000000  ENIOR      = 1  ENIOW= 1
  IUSTOP = 1     CLKSTOP = 0
```

NAME

bps – Sets and enables breakpoint actions

SYNOPSIS

bps i|c|ic, p|p1|p2|or|ir|ow|iw, *address1*, [*address2*]

DESCRIPTION

This command sets and enables breakpoint actions and specifies addresses to be used in address compare operations.

i	Stops instruction issue on any address match.
c	Stops the clock on any address match.
ic	Both of the above.
p	Enables comparison of the program counter with both **PADR1** and **PADR2**. Both **PADR1 and PADR2** are set to *address1*. It is invalid syntax to specify *address2*.
p1	Enables comparison of the program counter with both **PADR1** and **PADR2**. Only **PADR1** is set to *address1*. **PADR2** remains unchanged. It is invalid syntax to specify *address2*.
p2	Enables comparison of the program counter with both **PADR1** and **PADR2**. Only **PADR2** is set to *address1*. **PADR1** remains unchanged. It is invalid syntax to specify *address2*.
or	Enables address comparison if operand fetches using memory ports 0, 1, and 2 for falling between **OPNDADRL** and **OPNDADRH**. **OPNDADRL** is set to *address1*. **OPNDADRH** is set to *address2* if specified; otherwise, to *address1*.
ir	Enables address comparison if the address of an I/O fetch using memory port 2 falls between **IOADRL** and **IOADRH**. **IOADRL** is set to *address1*. **IOADRH** is set to *address2* if specified; otherwise, to *address1*.
ow	Enables address comparison if operand writes using memory port 3 for falling between **OPNDADRL** and **OPNDADRH**. **OPNDADRL** is set to *address1*. **OPNDADRH** is set to *address2* if specified, otherwise, to *address1*.
iw	Enables address comparison if I/O writes using memory port 3 for falling between **IOADRL** and **IOADRH**. **IOADRL** is set to *address1*. **IOADRH** is set to *address2*, if specified, otherwise, to *address1*.

EXAMPLES

The following command line sets the breakpoint and enables address comparison. If operand writes using memory port 3 fall between central memory word 1008 hexadecimal and 2000 hexadecimal, then instruction issue stops.

```
bps i,ow,1008,2000
```

NOTES

IU must be OFF or RST must be ON.

Fields description. The parameters in the first field following the command operator determine the action taken at the break point. The parameters in the second field specify the condition for the breakpoint to occur. The third field specifies the memory address used for the compare.

NAME

cat – Displays the contents of a file on the SCSI disk

SYNOPSIS

cat [-*n*] *filename*

DESCRIPTION

-*n*	Outputs each line with the line number and byte count of the first byte.

NAME

cd – Changes the current directory on the hard disk

SYNOPSIS

cd *path*

DESCRIPTION

path	Absolute or relative path name of the desired directory.

EXAMPLES

Changes the current directory to the root directory.

```
cd  /
```

Changes the current directory to the subdirectory **BOOT**.

```
cd  BOOT
```

Changes the current directory to the directory **TEST** from any other directory.

```
cd /TEST
```

NAME

clk – Turns the clock on or off, or steps the clock

SYNOPSIS

clk [**on**|**off**|*n*]

DESCRIPTION

The **clk** command starts or stops the mainframe CPU or steps the clock a specified number of times if the clock is already off.

on	Turns the clock on.
off	Turns the clock off.
n	Specifies the number of times to step the clock if the clock is off. If the clock is on, an error message will be issued.

NOTES

If no parameter is specified and the CPU clock is off, the clock is stepped once. An error message will be issued if the CPU clock is currently on and no parameter is specified.

A **clk on** is executed automatically when the IOS first goes to multitasking mode.

NAME

conswitch – Toggles console from IOS to UNICOS system console

SYNOPSIS

conswitch

DESCRIPTION

Only executable from the IOS, the **conswitch** command does the equivalent of a <CNTRL>A to toggle the console terminal from acting as the IOS console to the UNICOS console interface. This command is primarily used in scripts to automate the transfer of control from the IOS to UNICOS.

NAME

cmt – Compares memory text

SYNOPSIS

cmt *filespec*

DESCRIPTION

The **cmt** command compares the content of the CRAY XMS system memory with the content of a named file. The file extension must be **.CMP**.

filespec	The name of the text file to be used for the comparison.

EXAMPLES

The following command line reads the **bas2x.cmp** file, from the hard disk, and compares the contents of central memory words with those specified in the file.

```
cmt bas2x.cmp
```

NOTES

This command accesses central memory through the data channels; therefore, the CPU clock must be on.

NAME

cmts – Compares memory text using scan mode

SYNOPSIS

cmts *filespec*

DESCRIPTION

The **cmts** command compares the content of the CRAY XMS system memory with the content of a named file. The file extension must be **.CMP**

filespec The name of the text file to be used for the comparison.

EXAMPLES

The following command reads the **bas2x.cmp** file from the hard disk, and compares the contents of central memory words with those specified in the file.

```
cmts bas2x.cmp
```

NOTES

This command accesses central memory by scanning; therefore, the CPU clock must be off.

NAME

count – Enables counter

SYNOPSIS

count *init*|*inc*|*print*

DESCRIPTION

The **count** command enables a counter that counts the number of passes that have been executed if a loop is used.

init	Initializes the counter to 0
inc	Increments the counter by 1
print	Prints the current value of the counter

EXAMPLES

The following command line displays the count on the terminal screen in decimal:

```
count print
```

NOTES

This command only executes in a shell script.

NAME

cp – Makes a copy of a file

SYNOPSIS

cp *filespec1 filespec2*

DESCRIPTION

filespec1	File specification of the source file.
filespec2	File specification of the destination file.

EXAMPLES

Copy all files in **test1** to **test 2**:

```
cp test1/*.*  test2/*.*
```

Copy all files from directory **usr/type** to **usr1/type** regardless of the current directory:

```
cp /usr/type/ *.*  usr1/type/ *.*
```

CAUTION

Destination files will be overwritten if they already exist.

NAME

dadisable – Disables defective drives in a disk array

SYNOPSIS

dadisable **p***cd* *unit*

DESCRIPTION

This command disables bad drive in disk array and substitutes with hot standby drive if it is available.

p*cd*	Physical controller/device number for the target bank. This parameter is specified in the form **p***cd* where:
	p Mandatory
	c Physical controller number. Valid numbers are 8 to B hexadecimal.
	d Physical device or bank number. Valid numbers are 0 through 3.
unit	Physical disk number in the target bank. Unit numbers 0 through 8 are valid.

EXAMPLES

If there is no stand-by drive in disk array and you type:

```
RT>dadisable p80 4
```

You receive the following:

```
Drive 4 may now be physically replaced.

RT>
```

If there is a stand-by drive in disk array and you type:

```
RT>dadisable p80 4
```

You receive the following:

```
Do you want to substitute the stand-by drive? (y/n) y

Reconstructing data on standby drive.  Please wait.
```

While processing, the system will output periods (dots) at intervals to show the process is alive, and an RT> prompt to indicate completion:

```
. . . . . . . . . . . . . . . .  . . . . . . . . . .

Drive 4 may now be physically replaced.

RT>
```

NOTES

This command has two functions: it disables a bad drive and assigns a stand-by drive. If you did not assign the stand-by drive when you disabled the bad drive, you can reissue this command to assign stand-by drive.

Each dot after command issued represents 1 minute.

NAME

daflwr – Disk array flaws read

SYNOPSIS

daflawr p*cb d filespec*

DESCRIPTION

Reads the Raw Flaw table for the given physical disk and places the information in the given file in a fixed format. This format is the same as is expected by **DAFLAWW**.

p*cb*	Physical controller/bank number for the target drive. This parameter is specified in the form **p***cb* where:
	p Is mandatory.
	c Is the physical controller number. Valid numbers are 8 to B hexadecimal.
	b Is the physical bank number. Valid numbers are 0 through 3.
d	Physical disk number of the target drive. Valid drive numbers are 0 through 9.
filespec	The name of the file to be created for the flaw information. This parameter must be in standard console format.

EXAMPLES

The following command line will read the flaw map from the ast data drive on controller 8/bank 1. A formatted copy of the map will be written to the hard disk in a file named **001793.s–1DAF**.

```
daflawr  p81 7 001793.daf
```

NAME

daflaww – Disk array flaws write

SYNOPSIS

daflaww p*cb d filespec*

DESCRIPTION

The **daflaww** command scans the given file for flaw entries then writes them into the Growth Error table for the given physical disk. The file is expected to be in the following format:

Lines in the file beginning with the character # are ignored. All other lines should have four hex fields starting in the first column. The four fields are as follows:

Cylinder Head Position Bit-Length

p*cb*	Physical controller/bank number for the target drive. This parameter is specified in the form **p***cb* where:

p	Mandatory.
c	Physical controller number. Valid numbers are 8 to B hexadecimal.
b	Physical bank number. Valid numbers are 0 through 3.
d	Physical disk number of the target drive. Valid drive numbers are 0 to 9.

filespec	The name of the file to be read for the flaw information. This parameter must be in standard console format.

RESPONSE

This command will finish with the following message where the value *N* is given in hexidecimal:

```
N flaws added to the GET.
```

EXAMPLES

The following command will read flaw information from a file named **000762.DAF**. The flaw entries will be added to the Growth Error table for the parity drive on controller 9/bank 1.

```
daflaww  p91 8 000762.daf
```

NOTES

The file naming convention is 12345678.DAF where 1-8 are the last eight numbers of the given disk's serial number and DAF stands for disk array flaws.

NAME

daformat – Disk array format command

SYNOPSIS

daformat P*cd* **B***xxx* [*level*]

DESCRIPTION

Th **daformat** command formats the specified disk(s) at the level requested.

P	Mandatory.
c	Controller number; 8 to B
d	Bank number; 0 to 3
B	Mandatory.
xxx	Bit map that indicates which drives on the given controller/bank are to be formatted. This is a 10 bit, right-to-left bit map. This valid values are 1 to 3FF hexidecimal. The right-most bit indicates drive 0, and the left-most bit (200 hex) indicates drive 9.
level	Integer format level. Level 1 initializes the system area; 2 reads the media defect list and initializes the system area; 3 discards the Growth Error table (GET) and formats the user area; 4 merges the GET with the Raw Flaw table (RFT) and formats the user area; and 5 (default) will merge the GET with the RFT and format the bad tracks only, preserving user data.

EXAMPLES

The following command formats all disks on the DAS destroying any data:

```
daformat  p80  b3ff  4
```

Either of the following commands formats all disks on the DAS preserving data:

```
daformat  p80  b3ff

daformat  p80  b3ff  5
```

NOTES

It is recommended that **daverify** be run to add any bad sectors to the GET before running **daformat**. It is also recommended that **daformat** be run with no level specified (defaults to nondestructive level 5).

SEE ALSO

daverify

NAME

dareplace – Reconstructs data on replaced disk

SYNOPSIS

dareplace p*cd unit*

DESCRIPTION

The **dareplace** command reconstructs data onto newly replaced disk.

p*cd*	Physical controller/device number for the target bank. This parameter is specified in the form **p***cd* where:
	p Mandatory
	c Physical controller number
	d Physical device or bank number
	Controller numbers 8 to B hexadecimal and bank numbers 0 through 3 are valid.
unit	Physical disk number in the target bank. Unit numbers 0 through 8 are valid.

EXAMPLES

If controller 8, bank 0, drive 5 was disabled, after physically replacing a formatted drive, you should type the following:

```
dareplace P80 5
Reconstructing data on replacement drive.  Please wait.
```

While processing, the system will output periods (dots) at intervals to show the process is alive, and a hyphen prompt to indicate completion:

```
. . . . . . . . . . . . . . . . . . . . . .   .

RT>
```

NOTES

A formatted disk must be replaced after **DADISABLE** and before **DAREPLACE**.

Before issuing this command, issue the **DAINFO** command to make sure the drive has either been disabled or substituted.

Each dot after command issued represents one minute.

NAME

daverify – Scans drive to verify media

SYNOPSIS

daverify P*cd* **B***xxx* [*level*]

DESCRIPTION

The **daverify** command verifies the media at the level specified.

P	Mandatory.
c	Controller number; 8 to B
d	Bank number; 0 to 3
B	Mandatory.
xxx	Bit map that indicates which drives on the given controller/bank are to be formated. This is a 10 bit, right-to-left bit map. This valid values are 1 to 3FF hex. The right-most bit indicates drive 0, and the left-most bit (200 hex) indicates drive 9.
level	Integer scrub level. Level 0 does a read check only, data is preserved; level 1 writes the pattern 00000000h then a read check; level 2 writes the pattern FFFFFFFFh then a read check; level 3 writes the pattern AAAAAAAAh then a read check; level 4 writes the pattern 55555555h then a read check; level 5 writes the pattern CCCCCCCCh then a read check; level 6 writes the pattern 33333333h then a read check; level 7 writes the pattern 6DB6DB6Dh then a read check; level 8 writes the pattern 92492492h then a read check; level 9 writes the pattern C6DEC6DEh then a read check; level A writes the pattern 39213921h then a read check; level B..E reserved (0's); and level F writes random then read check (recommended run several times).

EXAMPLES

The following command simply scans the disk updating the GET with errors:

```
daverify  p80  b3ff
```

The following command patterns the disk:

```
daverify  p80  b3ff 6
```

NOTES

It is recommended that **daverify** be run with no level specified (default) to simply read the disk and update the Growth Error table (GET). This could be followed by a **daformat** (default level 5) to incorporate the errors found into the Raw Flaw table which is used to map around bad sectors.

SEE ALSO

daformat

NAME

dawconfig – Disk array write configuration command

SYNOPSIS

dawconfig **p***c* *filespec*

DESCRIPTION

The **dawconfig** command reads configuration parameters from the given file and writes them to the selected disk array controller.

p*c* Physical controller number. This parameter is specified in the form **p***c* where:

- **p** Mandatory
- *c* Physical controller number. Controller numbers 8 to B hexidecimal are valid.

filespec The name of the file containing the configuration information. This parameter must be in standard console format.

EXAMPLES

Either of the following command lines sends the configuration parameters in file **esdi4k8.das** to controller 8:

```
dawconfig  p8 esdi4k8

dawconfig  p8 esdi4k8.das
```

Either of the following commands sends the configuration parameters in file **smd4k4.das** to controller A:

```
dawconfig  pa smd4k4

dawconfig  pa smd4k4.das
```

NOTES

This command must be followed by a controller reset or an IOS reset to activate the new configuration parameters.

The format of the ASCII input file is as follows:

- All lines which begin with the character # are ignored as comment lines.
- A line beginning with the character + followed by 3 bytes separated by spaces must be given to set the **SEL**, **TAB**, and **offset** address parameters respectively.
- The next line contains up to 16 bytes separated by spaces. This is taken as configuration data and it is written to the previously given **SEL/TAB/offset** address. Only the given bytes are written.
- An address line must precede each data line.
- All address and data values are given in hexidecimal.

- There are three standard configuration files. They set the configuration for the three standard combinations of ESDI or SMD drives configured for 4 or 8 data drives. All are set to give a 4 Kbyte block size. The files are as follows:

  esdi4k8.das
  esdi4k4.das
  smd4k4.das

NAME

ddflawr – Reads factory flaw map and writes it to a file

SYNOPSIS

ddflawr p*mn* [*sn*]

DESCRIPTION

The **ddflawr** command reads the factory flaw map of an ESDI system drive and writes it to a file on the IOS local disk.

p	Mandatory
m	Controller number, 2 to 6 hexidecimal
n	Drive number, 0 to 3
sn	Serial number of the drive to be formatted

NAME

ddformat – Formats the system disk

SYNOPSIS

ddformat p*mn*

DESCRIPTION

The **ddformat** command formats an ESDI system disk. It will rewrite each sector header of the entire drive.

p	Mandatory
m	Controller number 0 and 1 not used
	0 to 3 for ESDI drive
n	Drive number 0 and 1 not used
	0 to 3 for ESDI drive

NOTES

This process runs in the background; therefore, it appears to complete instantaneously. If the format fails the following message will be displayed:

```
format controller m, drive n, error = #
```

CAUTIONS

All data on the disk will be lost during the formatting process.

NAME

ddtest – Ciprico disk controller self test

SYNOPSIS

ddtest Pc [–*xx*]

DESCRIPTION

The **ddtest** command involves the self test function on the selected disk controller.

01H	Tests scratchpad RAM
02H	Tests cache RAM
04H	Checksum firmware PROM
08H	Tests nonmemory hardware
7FH	All the above tests

NAME

debug – Sets a Debug flag

SYNOPSIS

debug [*n*]

DESCRIPTION

This command sets a Debug flag (PDEBUG) in the IOS.

n Value of the (PDEBUG) flag. If no parameter is entered, this command displays the current value of the flag.

NAME

dioq – Displays IOQ

SYNOPSIS

dioq

DESCRIPTION

The **dioq** command is at the scan level; therefore, the clock must be off. These are scan out only flip-flops. This command is of importance to hardware engineers only.

NAME

dlba – Displays the last branch address

SYNOPSIS

dlba

DESCRIPTION

The **dlba** command displays the last branch address taken by the mainframe CPU.

EXAMPLES

```
                                   P      SEQ #

        LBA0 (most recent)       5D5D5D   D

        LBA1                     5D5D5D   D

        LBA2                     5D5D5D   D

        LBA3                     5D5D5D   D

        LBA4                     5D5D5D   D

        LBA5                     5D5D5D   D

        LBA6                     325D5D   D

        LBA7                     5D5D5D   D
```

NAME

dm – Displays memory

SYNOPSIS

dm [*start*], [*count*], [**p**], [**h**|**o**|**a**] , [**h**|**o**|**a**]

DESCRIPTION

This command displays the contents of central memory.

start	Memory is displayed starting at the specified relative address. (The default is the base address the first time, or the last address displayed +1.)
count	Number of words to be displayed. (Default: 16 for the first time, or the last count issued.)
p	If the value of the program counter is specified, start is taken as the parcel address.
h	Memory is displayed in hexadecimal. (Default.)
o	Memory is displayed in octal.
a	Memory is displayed in ASCII.

EXAMPLES

The following command line displays 24 words of central memory starting at 100 hexadecimal.

```
dm 100,24
```

NOTES

Memory contents can be displayed in two radices. If the specified radices are the same, the memory contents are displayed in only one. This does not change the radices permanently. To change the radices permanently, use the **mode** command.

See the **base** command for more information on the start address.

This command accesses central memory through the data channels; therefore, the CPU clock must be on.

NAME

dms – Displays memory using scan mode

SYNOPSIS

dms [*start*] , [*count*], [**p**], [**h**|**o**|**a**], [**h**|**o**|**a**]

DESCRIPTION

start	Memory is displayed starting at the specified relative address. (The default is the base address the first time, or the last address displayed +1.)
count	Number of words in decimal to be displayed. (Default: 16 for the first time, or the last count issued.)
p	If the value of the program counter is specified, start is taken as the parcel address.
h	Memory is displayed in hexadecimal. (Default)
o	Memory is displayed in octal.
a	Memory is displayed in ASCII.

EXAMPLES

The following command line displays 24 words of central memory starting at 100 hexadecimal:

```
dms 100,24
```

NOTES

Memory contents can be displayed in two radices. If the specified radices are the same, the memory contents are displayed in only one. This does not change the radices permanently. To change the radices permanently, use the **mode** command.

See the **base** command for more information on the start address.

This command accesses central memory by scanning; therefore, the CPU clock must be off.

NAME

dr – Displays the value of the program counter P or the state of the pmatched flag

SYNOPSIS

dr p|pm

DESCRIPTION

p	Displays the value of the program counter P
pm	Displays the value of the pmatched flag

EXAMPLES

The following command line displays the value of the program counter:

```
dr  p
```

NAME

dreg – Dumps registers

SYNOPSIS

dreg [*regname*]

DESCRIPTION

The **dreg** command dumps the CPU registers to the IOS screen.

regname	Name of the register to be displayed. (Default: a, b, and s registers for the first time, or the last registers displayed.) Valid register names are: a, s, b, t, v0-7, sm, sb, st, i0-3.

NAME

dump – Captures mainframe memory

SYNOPSIS

dump [–v] **p***cd sa* [*word_count*]

DESCRIPTION

The **dump** command starts at address zero in the mainframe memory and copies 'word count' (default 16 mega-words) words to a system disk in a format that **cpdmp**(1M) recognizes. The system disk partition that is the target for the dump must have been first initialized with the **idmp**(1M) command. Upon rebooting the mainframe, the dump may be recovered using **cpdmp** and examined by **crash**(1M).

–v	Verbose; outputs the dump header structure from disk, etc.
p	Required designator letter indicating the start of a system disk definition
c	Controller number of system disk containing dump partition
d	Device(disk) number of system disk containing dump partition
sa	Starting sector address of dump partition
word_count	Optional word count to save of mainframe memory. (Default: 16MW.)

EXAMPLES

In the following example, a system dump is placed on controller 8 (DAS), disk 0, starting at sector 0:

```
    dump p80 0
```

NOTES

Normally, this command will be executed from the **sysdump** script which should be modified at installation time to reference the correct disk and sector addresses of the dump partition.

Currently, this command requires that the dump partition is a contiguous partition on disk.

This command accesses central memory through the data channels; therefore, the CPU clock must be on.

SEE ALSO

cpdump(1M), **idmp**(1M), **crash**(1M)

NAME

dxp – Gives a formatted display of an exchange package

SYNOPSIS

dxp [*addr*]

DESCRIPTION

addr	Relative central memory address of the desired exchange package. (The default is the base address.)

NOTES

See the **base** command for more information on the *addr* address.

This command accesses central memory through the data channels; therefore, the CPU clock must be on.
base()

NAME

dxps – Displays an exchange package using scan mode

SYNOPSIS

dxps [*addr*]

DESCRIPTION

The **dxps** command displays an exchange package, using scan mode.

addr Relative central memory address of the desired exchange package. (The default is the base address.)

NOTES

See the **base** command for more information on the *addr* address.

This command accesses central memory by scanning; therefore, the CPU clock must be off.

SEE ALSO

base

NAME

echo – Displays a message

SYNOPSIS

echo *string*

DESCRIPTION

string Character string. This string will be displayed on the screen when the command executes.

EXAMPLES

The following line will print the message **Debug Test Message** when the command file executes:

```
echo  Debug Test Message
```

NAME

ed – Text editor

SYNOPSIS

ed *file*

DESCRIPTION

The **ed** editor is the standard text editor. If the *file* argument is given, **ed** simulates an *e* command (see the following text) on the named file; that is to say, the file is read into **ed**'s buffer so that it can be edited.

The **ed** editor operates on a copy of the file it is editing; changes made to the copy have no effect on the file until a **w** (write) command is given. The copy of the text being edited resides in a temporary file called the *buffer*. There is only one buffer.

Commands to **ed** have a simple and regular structure: zero, one, or two *addresses* followed by a single-character *command*, possibly followed by parameters to that command. These addresses specify one or more lines in the buffer. Every command that requires addresses has default addresses, so that the addresses can very often be omitted.

In general, only one command may appear on a line. Certain commands allow the input of text. This text is placed in the appropriate place in the buffer. While **ed** is accepting text, it is said to be in *input mode*. In this mode, *no* commands are recognized; all input is merely collected. Input mode is left by typing a period (**.**) alone at the beginning of a line, followed immediately by a carriage return.

The **ed** editor supports a limited form of *regular expression* notation; regular expressions are used in addresses to specify lines and in some commands (e.g., **s**) to specify portions of a line that are to be substituted. A regular expression (RE) specifies a set of character strings. A member of this set of strings is said to be *matched* by the RE. The REs allowed by **ed** are constructed as follows:

The following *one-character REs* match a *single* character:

1.1 An ordinary character (*not* one of those discussed in 1.2 below) is a one-character RE that matches itself.

1.2 A backslash (\) followed by any special character is a one-character RE that matches the special character itself. The special characters are:

a. **.**, *****, **[**, and \ (period, asterisk, left square bracket, and backslash, respectively), which are always special, *except* when they appear within square brackets (**[ ]**; see 1.4 below).

b. **^** (caret or circumflex), which is special at the *beginning* of an *entire* RE (see 3.1 and 3.2 below), or when it immediately follows the left of a pair of square brackets (**[ ]**) (see 1.4 below).

c. **$** (dollar sign), which is special at the *end* of an entire RE (see 3.2 below).

d. The character used to bound (such as, delimit) an entire RE, which is special for that RE [for example, see how slash (/) is used in the *g* command, below.]

1.3 A period (**.**) is a one-character RE that matches any character except new-line.

1.4 A non-empty string of characters enclosed in square brackets (**[ ]**) is a one-character RE that matches *any one* character in that string. If, however, the first character of the string is a circumflex (**^**), the one-character RE matches any character *except* new-line and the remaining characters in the string. The **^** has this special meaning *only* if it occurs first in the string. The minus (–) may be used to indicate a range of consecutive ASCII characters; for example, **[0–9]** is equivalent to **[0123456789]**. The – loses this special meaning if it occurs first (after an initial **^**, if any) or last in the string. The right square bracket (**]**) does not terminate such a string when it is the first character within it (after an initial **^**, if any); e.g., **[ ]a–f]** matches either a right square bracket (**]**) or one of the letters **a** through **f** inclusive. The four characters listed in 1.2.a above stand for themselves within such a string of characters.

The following rules may be used to construct *RE*s from one-character REs:

2.1 A one-character RE is a RE that matches whatever the one-character RE matches.

2.2 A one-character RE followed by an asterisk (*) is a RE that matches *zero* or more occurrences of the one-character RE. If there is any choice, the longest leftmost string that permits a match is chosen.

2.3 A one-character RE followed by \{*m*\}, \{*m,*\}, or \{*m,n*\} is a RE that matches a *range* of occurrences of the one-character RE. The values of *m* and *n* must be non-negative integers less than 256; \{*m*\} matches *exactly m* occurrences; \{*m,*\} matches *at least m* occurrences; \{*m,n*\} matches *any number* of occurrences *between m* and *n* inclusive. Whenever a choice exists, the RE matches as many occurrences as possible.

2.4 The concatenation of REs is a RE that matches the concatenation of the strings matched by each component of the RE.

2.5 A RE enclosed between the character sequences \(and \) is a RE that matches whatever the unadorned RE matches.

2.6 The expression *n*, matches the same string of characters as was matched by an expression enclosed between \(and \) *earlier* in the same RE. Here *n* is a digit; the subexpression specified is that beginning with the *n*th occurrence of \(counting from the left. For example, the expression **^\(.*\)\1$** matches a line consisting of two repeated appearances of the same string.

Finally, an *entire RE* may be constrained to match only an initial segment or final segment of a line (or both).

3.1 A circumflex (^) at the beginning of an entire RE constrains that RE to match an *initial* segment of a line.

3.2 A dollar sign ($) at the end of an entire RE constrains that RE to match a *final* segment of a line.

The construction ^*entire RE*$ constrains the entire RE to match the entire line.

The null RE (such as, //) is equivalent to the last RE encountered. See also the last paragraph before *FILES* below.

To understand addressing in **ed** it is necessary to know that at any time there is a *current line*. Generally speaking, the current line is the last line affected by a command; the exact effect on the current line is discussed under the description of each command. *Addresses* are constructed as follows:

1. The character **.** addresses the current line.
2. The character **$** addresses the last line of the buffer.
3. A decimal number *n* addresses the *n*th line of the buffer.
4. '*x* addresses the line marked with the mark name character *x*, which must be a lowercase letter. Lines are marked with the *k* command described below.
5. A RE enclosed by slashes (/) addresses the first line found by searching *forward* from the line *following* the current line toward the end of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the beginning of the buffer and continues up to and including the current line, so that the entire buffer is searched. See also the last paragraph before *FILES*.
6. A RE enclosed in question marks (?) addresses the first line found by searching *backward* from the line *preceding* the current line toward the beginning of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the end of the buffer and continues up to and including the current line. See also the last paragraph before *FILES* below.

7. An address followed by a plus sign (+) or a minus sign (–) followed by a decimal number specifies that address plus (respectively minus) the indicated number of lines. The plus sign may be omitted.
8. If an address begins with + or –, the addition or subtraction is taken with respect to the current line; e.g, **–5** is understood to mean **.–5**.
9. If an address ends with + or –, then 1 is added to or subtracted from the address, respectively. As a consequence of this rule and of Rule 8, immediately above, the address – refers to the line preceding the current line. (To maintain compatibility with earlier versions of the editor, the character ^ in addresses is entirely equivalent to –.) Moreover, trailing + and – characters have a cumulative effect, so – – refers to the current line less 2.
10. For convenience, a comma (**,**) stands for the address pair **1,$**, while a semicolon (**;**) stands for the pair **.,$**.

Commands may require zero, one, or two addresses. Commands that require no addresses regard the presence of an address as an error. Commands that accept one or two addresses assume default addresses when an insufficient number of addresses is given; if more addresses are given than such a command requires, the last one(s) are used.

Typically, addresses are separated from each other by a comma (**,**). They may also be separated by a semicolon (**;**). In the latter case, the current line (**.**) is set to the first address, and only then is the second address calculated. This feature can be used to determine the starting line for forward and backward searches (see Rules 5 and 6, above). The second address of any two-address sequence must correspond to a line that follows, in the buffer, the line corresponding to the first address.

In the following list of **ed** commands, the default addresses are shown in parentheses. The parentheses are .I not part of the address; they show that the given addresses are the default.

It is generally illegal for more than one command to appear on a line. However, any command (except **e**, **f**, **r**, or **w**) may be suffixed by **l**, **n**, or **p** in which case the current line is either listed, numbered, or printed, respectively, as discussed below under the **l**, **n**, and **p** commands.

(.) a
<text>
.

The **append** command reads the given text and appends it after the addressed line; **.** is left at the last inserted line, or, if there were none, at the addressed line. Address 0 is legal for this command: it causes the appended text to be placed at the beginning of the buffer. The maximum number of characters that may be entered from a terminal is 256 per line (including the new-line character).

(.) c
<text>
.

The **change** command deletes the addressed lines, then accepts input text that replaces these lines; **.** is left at the last line input, or, if there were none, at the first line that was not deleted.

(. , .) d

The **delete** command deletes the addressed lines from the buffer. The line after the last line deleted becomes the current line; if the lines deleted were originally at the end of the buffer, the new last line becomes the current line.

e *file*

The **edit** command causes the entire contents of the buffer to be deleted, and then the named file to be read in; **.** is set to the last line of the buffer. If no file name is given, the currently-remembered file name, if any, is used (see the **f** command). The number of characters read is typed; *file* is remembered for possible use as a default file name in subsequent **e**, **r**, and **w** commands. If *file* is replaced by **!**, the rest of the line is taken to be a shell [*sh*(1)] command whose output is to be read. Such a shell command is *not* remembered as the current file name. See also *DIAGNOSTICS* below.

E *file*

The Edit command is like **e** , except that the editor does not check to see if any changes have been made to the buffer since the last **w** command.

f *file*

If *file* is given, the file name command changes the currently-remembered file name to *file*; otherwise, it prints the currently-remembered file name.

(1 , $)g/ *RE* / "*command list*

In the global command, the first step is to mark every line that matches the given RE. Then, for every such line, the given *command list* is executed with **.** initially set to that line. A single command or the first of a list of commands appears on the same line as the global command. All lines of a multiline list except the last line must be ended with a \; **a**, **i**, and **c** commands and associated input are permitted. The **.** terminating input mode may be omitted if it would be the last line of the *command list*. An empty *command list* is equivalent to the **p** command. The **g**, and **v** commands are *not* permitted in the *command list*. See also *BUGS* and the last paragraph before *FILES*. (.) i

<text>
.

The insert command inserts the given text before the addressed line; **.** is left at the last inserted line, or, if there were none, at the addressed line. This command differs from the **a** command only in the placement of the input text. Address 0 is not legal for this command. The maximum number of characters that may be entered from a terminal is 256 per line (including the new-line character).

(.,.+1)j

The **join** command joins contiguous lines by removing the appropriate new-line characters. If exactly one address is given, this command does nothing.

(.)k*x*

The **mark** command marks the addressed line with name **x** , which must be a lowercase letter. The address '*x* then addresses this line; **.** is unchanged.

(.,.)l

The **list** command prints the addressed lines in an unambiguous way: a few nonprinting characters (for example, *tab, backspace*) are represented by visually mnemonic overstrikes. All other nonprinting characters are printed in octal, and long lines are folded. An **l** command may be appended to any command other than **e**, **f**, **r**, or **w**.

(.,.)m*a*

The **move** command repositions the addressed line(s) after the line addressed by **a**. Address 0 is legal for **a** and causes the addressed line(s) to be moved to the beginning of the file. It is an error if address **a** falls within the range of moved lines; **.** is left at the last line moved.

(.,.)p

The **print** command prints the addressed lines; **.** is left at the last line printed. The **p** command may be appended to any other command other than **e**, **f**, **r**, or **w** . For example, *dp* deletes the current line and prints the new current line.

P

The editor will prompt with a * for all subsequent commands. The **P** command alternately turns this mode on and off; it is initially off.

q

The **quit** command causes **ed** to exit. No automatic write of a file is done; however, see *DIAGNOSTICS*.

Q

The editor exits without checking if changes have been made in the buffer since the last **w** command.

(\$)r*file*

The **read** command reads in the given file after the addressed line. If no file name is given, the currently-remembered file name, if any, is used (see **e** and **f** commands). The currently-remembered file name is *not* changed unless *file* is the very first file name mentioned since **ed** was invoked. Address 0 is legal for **r** and causes the file to be read at the beginning of the buffer. If the read is successful, the number of characters read is typed; . is set to the last line read in. If *file* is replaced by !, the rest of the line is taken to be a shell [*sh*(1)] command whose output is to be read. For example, "\$r !ls" appends current directory to the end of the file being edited. Such a shell command is not remembered as the current file name.

(.,.)s/*RE / replacement /* or
(.,.)s/*RE/replacement/***g** or
(.,.)s/*RE/replacement/***n** n = 1-512

The **substitute** command searches each addressed line for an occurrence of the specified RE. In each line in which a match is found, all (nonoverlapped) matched strings are replaced by the *replacement* if the global replacement indicator g appears after the command. If the global indicator does not appear, only the first occurrence of the matched string is replaced. If a number n appears after the command, only the *n*th occurrence of the matched string on each addressed line is replaced. It is an error for the substitution to fail on all addressed lines. Any character other than space or new line may be used instead of / to delimit the RE and the *replacement*; . is left at the last line on which a substitution occurred. See also the last paragraph before
FILES.

An ampersand (**&**) appearing in the *replacement* is replaced by the string matching the RE on the current line. The special meaning of **&** in this context may be suppressed by preceding it by \. As a more general feature, the characters *n*, where **n** is a digit, are replaced by the text matched by the *n*th regular subexpression of the specified RE enclosed between \(and \). When nested parenthesized subexpressions are present, *n* is determined by counting occurrences of \(starting from the left. When the character % is the only character in the *replacement*, the *replacement* used in the most recent substitute command is used as the *replacement* in the current substitute command. The % loses its special meaning when it is in a replacement string of more than one character or is preceded by a \.

A line may be split by substituting a new-line character into it. The new line in the *replacement* must be escaped by preceding it by \. Such substitution cannot be done as part of a **g** or **v** command list.

(.,.)t*a*

This command acts just like the **m** command, except that a *copy* of the addressed lines is placed after address **a** (which may be 0); . is left at the last line of the copy.

u

The **undo** command nullifies the effect of the most recent command that modified anything in the buffer, namely the most recent **a**, **c**, **d**, **g**, **i**, **j**, **m**, **r**, **s**, **t**, **v**, **G**, or **V** command.

(1,\$)v/*RE/command list*

This command is the same as the global command **g** except that the *command list* is executed with . initially set to every line that does *not* match the RE.

(1,\$)V/*RE /*

This command is the same as the interactive global command **G** except that the lines that are marked during the first step are those that do not match the RE.

(1,\$)w*file"*

The **write** command writes the addressed lines into the named file. If the file does not exist, it is created with mode 666 (readable and writable by everyone), unless your **umask** setting (see **umask**(1)) dictates otherwise. The currently-remembered file name is not changed unless *file* is the very first file name mentioned since **ed** was invoked. If no file name is given, the currently-remembered file name, if any, is used (see **e** and **f** commands); . is unchanged. If the command is successful, the number of characters written is typed. If *file* is replaced by **!**, the rest of the line is taken to be a shell (*sh*(1)) command whose standard input is the addressed lines. Such a shell command is *not* remembered as the current file name.

X

An encryption key is requested from the standard input. Subsequent **e**, **r**, and **w** commands will use this key to encrypt or decrypt the text (see **crypt**(1)). An explicitly empty key turns off encryption. Also, see the **–x** option of **ed**.

($)=

The line number of the addressed line is typed; . is unchanged by this command.

(.+1) <new-line>

An address alone on a line causes the addressed line to be printed. A new.line alone is equivalent to **.+1p**; it is useful for stepping forward through the buffer.

If an interrupt signal (ASCII DEL or BREAK) is sent, **ed** prints a **?** and returns to *its* command level.

Some size limitations: 512 characters per line, 256 characters per global command list, and 64 characters per file name. The limit on the number of lines depends on the amount of user memory: each line takes 1 word.

When reading a file, **ed** discards ASCII NUL characters. Files (e.g., **a.out**) that contain characters not in the ASCII set (bit 8 on) cannot be edited by **ed**.

If a file is not terminated by a new-line character, **ed** adds one and outputs a message explaining what it did.

If the closing delimiter of a RE or of a replacement string (such as, /) would be the last character before a new line, that delimiter may be omitted, in which case the addressed line is printed. The following pairs of commands are equivalent:

```
s/s1/s2  s/s1/s2/p
g/s1     g/s1/p
?s1      ?s1?
```

FILES

/tmp Default directory for temporary work file.

DIAGNOSTICS

?	For command errors or if a backspace is input (in which case you are left back in command mode).
?*file*	For an inaccessible file. (use the **help** and **Help** commands for detailed explanations).

If changes have been made in the buffer since the last **w** command that wrote the entire buffer, **ed** warns the user if an attempt is made to destroy **ed** 's buffer via the **e** or **q** commands. It prints ? and allows one to continue editing. A second **e** or **q** command at this point will take effect. The –s command-line option inhibits this feature.

WARNINGS

Reasonable editing sessions should be kept under 10 Kbytes. Lines are limited to 4096 characters.

When reading a file, **ed** discards ASCII NUL characters and all characters after the last new line. Files (such as **a.out**) that contain characters that are not in the ASCII set (bit 8 on) cannot be edited by **ed**.

Size limitations: Large files generate larger editor temporary files and cost many processor cycles on entry to **ed**.

NAME

errlog – Displays the IOS error log file in a readable format

SYNOPSIS

errlog [–*errlog file*]

DESCRIPTION

The **errlog** command reads the data file *errlog file* defaults to **adm/errlog** and enterprets the records for display. This file contains error entries reported from system peripherals and will go away in future releases.

NAME

exec – Executes a script

SYNOPSIS

exec [–*x*] *filename*

DESCRIPTION

The **exec** command interprets an ASCII file as IOS commands and executes each line of the specified file.

exec is invoked automatically by the IOS if the user enters the name of a file at the IOS command prompt.

–*x*	Debug flag; **exec** prints each line it is about to execute.

EXAMPLES

The following example interprets the file **/bin/boot**:

```
exec /bin/boot
```

NAME

fm – Fills memory

SYNOPSIS

fm *start*, *count*, [*parcelA*], [*parcelB*], [*parcelC*], [*parcelD*]

DESCRIPTION

The **fm** command fills memory with the specified values.

start	Relative address of memory to start filling.
count	Number of words (in decimal) to fill.
parcelA	Value to fill parcel A (most significant). (Default: 0.)
parcelB	Value to fill parcel B. (Default: 0.)
parcelC	Value to fill parcel C. (Default: 0.)
parcelD	Value to fill parcel D (least significant). (Default: 0.)

EXAMPLES

The following command line writes the value 123 5678 9ABC DEF0 to central memory word 100 hexadecimal through word 102 hexadecimal:

```
fm 100,3,123,5678,9ABC,DEF0
```

NOTES

At least 1 parcel has to be specified.

This command accesses central memory through the data channels; therefore, the CPU clock must be on.

NAME

fms – Fills memory using scan mode

SYNOPSIS

fms *start*, *count*, [*parcelA*], [*parcelB*], [*parcelC*], [*parcelD*]

DESCRIPTION

The **fms** command fills memory with the specified values.

start	Relative address of memory to start filling.
count	Number of words (in decimal) to fill.
parcelA	Value to fill parcel A (most significant). (Default: **0**.)
parcelB	Value to fill parcel B. (Default: **0**.)
parcelC	Value to fill parcel C. (Default: 0)
parcelD	Value to fill parcel D (least significant). (Default: **0**.)

EXAMPLES

The following command line writes the value 123 5678 9ABC DEF0 to central memory word 100 hexadecimal through word 102 hexadecimal:

```
fms 100,3,123,5678,9ABC,DEF0
```

NOTES

At least one parcel has to be specified.

This command accesses central memory by scanning; therefore, the CPU clock must be off.

NAME

fsf – Spaces forward one file on tape

SYNOPSIS

fsf [rst0]

DESCRIPTION

The **fsf** command spaces forward one file on tape.

rst0	Name of cartridge. (Default)

NAME

goto – Transfers control to the command pointed to by *label*

SYNOPSIS

goto **:***label*

DESCRIPTION

label	A string preceded by a colon (:), where the first eight characters are significant.

EXAMPLES

A command file containing the following three lines of code will print **Thanks a million** until interrupted by pressing <CTRL>C. This will kill any IOS command.

```
:AgainSam

msg Thanks a million

goto :AgainSam
```

NOTES

This command only executes in a shell script.

NAME

head – Displays first few lines of a specified file

SYNOPSIS

head [–*n*] *filename*

DESCRIPTION

The **head** command outputs the given number of lines (default 10) of the specified file.

–*n*	Specifies a line count

EXAMPLES

The following example displays the first 20 lines of the file **aaa**:

```
head -20 aaa
```

NAME

help – Lists the syntax for all commands

SYNOPSIS

help [*cmd*]

DESCRIPTION

The **help** command displays the syntax for all the commands or the specified command.

cmd	Name of the command to be displayed

NOTES

The **help** command will output all the commands that match its argument. For example, if you wanted a list of all commands that begin with the letter C, you would enter:

```
help C
```

EXAMPLES

A sample of the screen display is shown below.

```
AF EXT|OSC2|OSC3|PLL,n

AM address,[parcelA],[parcelB],[parcelC],[parcelD]

AP [page1],[page2]

AR [regname]

BASE [address]

.  .  .      (additional text not shown)

STEP n

TIME [dd/mm/yy],[hh:mm:ss]
```

NAME

if – Allows conditional transfer of control

SYNOPSIS

if *n* **goto** **:***label*

DESCRIPTION

The **if** command compares *n* with the return code from the previous command. If there is a match, control is transferred to the line immediately following the label.

n	Value to compare with the return code from the previous command
label	String preceded by a colon (:), where the first 8 characters are significant

EXAMPLES

A command file containing the following code will repeatedly read the value of the program counter and print it until it becomes equal to 1234. When the program counter equals 1234, the message **Done !!!** will be printed.

```
:KeepGoing

dr P

if 1234 goto :KeepGoing

msg Done !!!
```

NOTES

This command only executes in a shell script.

NAME

iosdump – Dumps the MIOP and IOBB memories to file on SCSI disk

SYNOPSIS

iosdump *filename*

DESCRIPTION

The **iosdump** command saves both the IOS processor's (MIOP) memory and the IOBB memory to the specified file on the SCSI hard disk.

This should be done if an IOS panic occurs.

NAME

iosinit – Initializes the CPU hardware after initial power-up

SYNOPSIS

iosinit

DESCRIPTION

The **iosinit** command sends initialization commands to the CSI board (scan path) inpreparation for communication between the IOS and the mainframe.

NOTES

This is done automatically by the IOS when it switches to multitasking mode.

NAME

iostart – Initiates communication between the I/O Subsystem and the operating system

SYNOPSIS

iostart

DESCRIPTION

iostart creates the tasks responsible for servicing the various requests between the operating system and its peripheral devices.

NAME

iostop – Stops communication between the IOS and UNICOS

SYNOPSIS

iostop

DESCRIPTION

The **iostop** command stops communication between the I/O Subsystem and the operating system. All executing commands and commands that are waiting to execute will be killed.

NOTES

This command does not stop the central CPU.

NAME

iu – Turns the instruction issue in the I-unit on or off

SYNOPSIS

iu on|off

DESCRIPTION

on	Resumes instruction issue
off	Stops instruction issue

NAME

lm – Transfers data

SYNOPSIS

lm p*cd*, *sa*, [*cma*], [*count*]

or

lm [*cma*], [*count*]

or

lm rst0, [*cma*], [*count*]

DESCRIPTION

The **lm** command transfers data from the specified system disk or tape drive to central memory. The data is transferred into central memory through the data channel.

p*cd*	Controller/drive number for a system disk. This parameter is specified in the form **p***cd* where **p** is the letter P, *c* is the physical controller number, and *d* is the physical drive number of the desired disk. Controller numbers 0 through 1 are reserved, 2 through 6 are used for ESDI disk controller and 8 to F are used for disk array controllers.
rst0	Name of cartridge tape. The letters **rst0** specify that the data is to be read from the cartridge tape named **rst0**.
sa	Sector address. This parameter must be specified in the case of transfers from a system disk; it specifies the starting logical sector address for the data. This parameter is ignored for tape sources; the read starts from the current tape position.
cma	Central memory address. This parameter specifies the starting central memory word address where the data is to be written. The default memory word is the current **base** value. (See **base** command.)
count	Word count. This parameter specifies the number of 64-bit words to write to central memory. This parameter must be specified for transfers from disk. In the case of transfers from tape, the transfer will continue until an EOF is detected unless the specified word count parameter is reached before the EOF.

EXAMPLES

The following command line transfers 1.3 million words of data from the ESDI disk on controller 2, unit 1, at the hexadecimal sector address 53BE to central memory at word address 100.

```
lm p21, 53be, 100, .1300000
```

This command will transfer data from the cartridge tape to central memory word address hexadecimal F00. (The read starts at the current tape position.)

```
lm rst0, , f00
```

NOTES

This command accesses central memory through the data channels; therefore, the CPU clock must be on.

The sector address specified in these commands assumes a sector length of 4096 bytes. The system console must compute the correct physical sector address based on the disk type.

In the case of a transfer from tape where an EOF is detected, the tape is left at the end of the block containing the EOF. Otherwise, the tape is left at the end of the block which held the last byte transferred. This means that a 1 word transfer will advance the tape 1 block. The block size is fixed at 64 central memory words (512 bytes) on the cartridge tape and is 512 central memory words (4096 bytes) on the 9-track tape.

SEE ALSO

base()

NAME

lmt – Loads memory text

SYNOPSIS

lmt *filespec*

DESCRIPTION

The **lmt** command loads the contents of a file to memory.

filespec Name of the file to be loaded. If no extension is given, the default **.xxx** will be assumed.

EXAMPLES

The following command line loads central memory with the contents of the hard disk file **bas2x.xxx**.

```
lmt bas2x.xxx
```

NOTES

This command accesses central memory through the data channels; therefore, the CPU clock must be on.

This command expects a text file format.

NAME

lmts – Loads memory text using scan mode

SYNOPSIS

lmts *filespec*

DESCRIPTION

The **lmts** command loads the contents of a file to memory using scan mode.

filespec	Name of the file to be loaded. If no extension is given, the default **.xxx** will be assumed.

EXAMPLES

The following command line loads central memory with the contents of the hard disk file **bas2x.xxx**:

```
lmts bas2x.xxx
```

NOTES

This command is at the scan level; therefore, the CPU clock must be off.

This command expects a text file format.

NAME

load – Loads IOS binary image into IOP (I/O processor)

SYNOPSIS

load [*–n*] [*filename*]

DESCRIPTION

The **load** command simply loads in a bootable image into the MIOP memory and attempts to boot from it. It accepts either a file or a device name as input.

If no file is specified, **load** looks for the file **/reboot** to exist. If it exists, **load** takes the contents of that file as the name of the file or device to boot from. If the **/reboot** file does not exist, **load** attempts to load in the file **/ios/ios** by default.

The **/reboot** file is created by the **reload** command.

–n Loads in the image but does not attempt to boot it.

EXAMPLES

The following example boots from the cartridge device:

```
load rst0
```

The following example boots the default IOS:

```
load
```

NAME

ls – Lists either all the directory entries, or only those for specified files

SYNOPSIS

ls [*path*] [*filename*[.*ext*]]

DESCRIPTION

[*path*] Path of directory to be listed. The default is the current directory.

[*filename*[.*ext*]]
File(s) to be listed. The default is all files will be listed.

NOTES

You can use the metacharacters ? and * in the file name and extension parameters.

NAME

mkdir – Makes a new directory on the hard disk

SYNOPSIS

mkdir [*path*/] *dirname*

DESCRIPTION

path	Path to the new directory. Optional if you are already in the path.
dirname	Name of the new directory.

EXAMPLES

To create a new directory called **test5** in the subdirectory **results** under the **root** directory enter the following:

```
mkdir results/test5
```

Another method would be to change directory to the **results** directory (using the **cd** command) and enter the following:

```
mkdir test5
```

NAME

mkfs – Makes file system

SYNOPSIS

mkfs [*drive*]

DESCRIPTION

drive	Specifies the drive to be formatted. If drive is omitted, the hard disk will be formatted.

CAUTION

Formatting the hard disk destroys all data contained on it.

EXAMPLES

```
mkfs c:
```

NAME

mm – Matches central memory

SYNOPSIS

mm *start*, *count*, [*parcelA*], [*parcelB*], [*parcelC*], [*parcelD*]

DESCRIPTION

The **mm** command matches central memory with the specified word.

start	Relative address of central memory to start matching
count	Number of central memory words to match
parcelA	Value to fill parcel A (most significant). (Default: 0.)
parcelB	Value to fill parcel B. (Default: 0.)
parcelC	Value to fill parcel C. (Default: 0.)
parcelD	Value to fill parcel D (least significant). (Default: 0.)

NOTES

This command acccsses central memory through the data channels; therefore, the CPU clock must be on.

NAME

mode – Changes radices used to display memory

SYNOPSIS

mode [h|o|a] [h|o|a]

DESCRIPTION

The **mode** command displays or changes the radices used to display the contents of memory. If no parameter is entered, the current setting is reported.

h	Memory is displayed in hexadecimal
o	Memory is displayed in octal
a	Memory is displayed in ASCII

The default bases are hexadecimal and octal.

EXAMPLES

The following command line sets the radix for addresses to hexadecimal and ASCII.

```
mode h a
```

NAME

more – Displays a file one screen at a time

SYNOPSIS

more *filename*

DESCRIPTION

The **more** command outputs a screenful at a time of the given file and then waits for any key input from the keyboard to continue.

The **more** command will quit upon receiving the **q** symbol.

EXAMPLES

The following example displays the file **aa**:

```
more aa
```

NAME

mount – Mounts local Winchester drive

SYNOPSIS

mount c:

DESCRIPTION

The **mount** command mounts, labels, and makes the Winchester drive available to the IOS.

This is done automatically at IOS boot time.

EXAMPLES

```
        mount c:
```

NAME

mv – Moves (renames) a file

SYNOPSIS

mv *filename1* *filename2*

DESCRIPTION

The **mv** command renames the file *filename1* to *filename2*. If *filename2* exists, it is deleted.

EXAMPLES

The following moves (or renames) file **a** to file **b**:

```
mv  a  b
```

NAME

pml – Performance monitor list

SYNOPSIS

pml

DESCRIPTION

The **pml** command displays the contents of the performance monitor counters.

EXAMPLES

```
GROUP = 1        ENMM = 1        ENCL   = 1                 RDINGPM = 1

                                 CLN # = 7

A) # of IO references                                :        0000000000

B) # OF IO conflicts                                 :        0000000000

C) # of scalar references                            :        0000000000

D) # of scalar conflicts                             :        0000000000

E) # of PORT0 block references                       :        0000000000

F) # of PORT1 block references                       :        0000000000

G) # of PORT2 block references                       :        0000000000

H) # of PORT3 block references                       :        0000000000

I) # of PORT0 block conflicts                        :        0000000000

J) # of PORT1 block conflicts                        :        0000000000

K) # of PORT2 block conflicts                        :        0000000000

L) # of PORT3 block conflicts                        :        0000000000

M) # of PORT3 vector references                      :        0000000000

N) # of instruction (4 words) fetches                :        0000000000
```

NAME

pms – Performance monitor set

SYNOPSIS

pms iu|mem, [*cln*], [*n*], [*mm*]

DESCRIPTION

This command activates performance monitor counters.

iu	Monitors instruction unit and functional unit activities.
mem	Monitors memory activities.
cln	Cluster number. Performance counters are active regardless of cluster number. However, if a cluster number is set, performance counters are only active if cluster number = *n*. If *cln* is specified, a corresponding *n* must also be specified.
n	Cluster number. Valid cluster numbers are 0 through 5. *n* should only be specified if *cln* is specified.
mm	Monitor mode. Performance counters are active only in user mode. However, if *mm* is set, performance counters are active in both user and monitor mode.

EXAMPLES

The following line activates the performance monitor counter for instruction unit activities when the cluster number is 3:

```
pms iu,cln,3
```

NAME

pwd – Displays the path name of the current working directory

SYNOPSIS

pwd

DESCRIPTION

The **pwd** command prints the path name of the working (current) directory.

NAME

reload – Initiates the reboot of the IOS

SYNOPSIS

reload [*filename*]

DESCRIPTION

If a file name (or device) is specified, **reload** creates a file called **/reboot** and places the file name into it.

The **reload** command then resets the VME which results in a reboot of the IOP from PROM.

If the autoboot switch is on, **load** is called from PROM and will boot the IOS from the file or device specified in the **/reboot** file or, by default, will boot /ios/ios.

If no file is specified, **load** attempts to load in the file /ios/ios by default.

EXAMPLES

The following example reboots from the cartridge device:

```
reload rst0
```

The following example simply reboots with the default IOS:

```
reload
```

NAME

rewind – Rewinds the cartridge tape

SYNOPSIS

rewind [rst0]

DESCRIPTION

rst0	Name of cartridge tape. This parameter specifies the cartridge tape.

NOTES

The default drive is **rst0** if no parameter is specified.

NAME

rm – Removes files and directories from the hard disk

SYNOPSIS

rm [**–r**] *file1* [*file2 file3* ...]

DESCRIPTION

The **rm** command removes any files listed on the command line. Directories are only removed if the **–r** option is specified.

For removing empty directories, see the **rmdir** command.

–r	Recursively remove directories.

EXAMPLES

The following example removes the file **aa** and the directory **/tmp/xx**:

```
rm -r aa /tmp/xx
```

SEE ALSO

rmdir

NAME

rmdir – Removes a directory

SYNOPSIS

rmdir [*path/*]*dirname*

DESCRIPTION

Removes a directory on the hard disk.

path	Path to the new directory.
dirname	Name of the new directory.

EXAMPLES

To remove a directory called **test5** in the subdirectory **results** under the root directory, enter the following:

```
rmdir results/test5
```

Another method would be to change directory to the **results** directory (using the **cd** command) and enter the following:

```
rmdir test5
```

NOTES

A subdirectory can only be removed if it is empty. That is, it contains only the special entries (.) and (..).

Only one subdirectory can be removed at a time.

The root directory and the current directory cannot be removed.

NAME

rst – Determines the reset state of the CPU

SYNOPSIS

rst off|on [, *filespec*]

DESCRIPTION

rst on forces the CPU into an architected reset state. **rst off** exits reset state which causes an exchange to central memory location 0 if the CPU clock is on.

off	Clears the reset line
on	Asserts the reset line
filespec	Asserts the reset line and loads the file specified. The default file name suffix of **.xxx** is used. *filespec* can only be specified if **on** is also specified.

NOTES

This command accesses central memory through the data channels; therefore the CPU clock must be on.

This command expects a text file format.

NAME

rsts – Determines the reset state of the CPU using scan mode

SYNOPSIS

rsts off|on [*filespec*]

DESCRIPTION

rsts on forces the CPU into an architected "reset" state. **rsts off** exits "reset" state, which causes an exchange to central memory location 0 if the CPU clock is on.

off	Clears the reset line
on	Asserts the reset line
filespec	Asserts the reset line and loads the file specified. The default file name suffix of **.xxx** is used. *filespec* can only be specified if **on** is also specified.

NOTES

This command accesses central memory by scanning; therefore the CPU clock must be off.

This command expects a text file format.

NAME

run – Loads and runs a file

SYNOPSIS

run *filespec*, [*seconds*]

DESCRIPTION

The **run** command loads and runs a file, then waits before stopping the clock. If a file with the same file name but with extension **.CMP** exists, the IOS will compare central memory with the contents of the file, using the I/O channel.

filespec	The name of the file to be loaded and executed. If *filespec* has no extension, the extension **.xxx** is appended.
seconds	The number of seconds to let the clock run. The default is 10.

NOTES

This command accesses central memory through the data channels; therefore, the CPU clock must be on.

NAME

runs – Loads and runs a file using scan path

SYNOPSIS

run *filespec*, [*seconds*]

DESCRIPTION

The **runs** command loads and runs a file, then waits before stopping the clock. If a file with the same file name, but with extension **.CMP** exists, the IOS will compare central memory with the contents of the file, using the scan path.

filespec	Name of the file to be loaded and executed. If *filespec* has no extension, the extension **.xxx** is appended.
seconds	Number of seconds to let the clock run. (The default is 10.)

NOTES

This command accesses central memory by scanning; therefore, the CPU clock must be off.

NAME

rupt – Interrupts CRAY XMS CPU from the console (MCU interrupt)

SYNOPSIS

rupt

DESCRIPTION

The **rupt** command generates an MCU interrupt from the IOS to the CRAY XMS CPU.

NOTES

The CPU clock must be on.

NAME

ser – Turns the serial mode on or off

SYNOPSIS

ser on|off

DESCRIPTION

The **ser** command turns the serial mode on or off.

on	No instruction overlapping during execution
off	Instruction overlapping enabled (normal execution)

NAME

set – Sets option for scripts

SYNOPSIS

set [–x]

DESCRIPTION

The **set** command currently only toggles the **–x** option for scripts which outputs each line prior to executing it from the script.

set with the **–x** argument turns this functionality on. **set** with no options turns it off.

–x	Turns on echoing of command lines prior to execution from script

NAME

sm – Transfers data from central memory to the specified system disk or tape drive

SYNOPSIS

sm p*cd*, *sa*, [*cma*], *count*

or

sm rst0 , , [*cma*], *count*

DESCRIPTION

If the data is being written to tape, an EOF is written following the data. The data is transferred from central memory through the data channel.

p*cd*	Controller/drive number for a system disk. This parameter is specified in the form **p***cd* where **p** is the letter p, *c* is the physical controller number, and *d* is the physical drive number of the desired disk. Controller numbers 0 to 1 are reserved, 2 through 6 are used for ESDI disk controller and 8 to F are used for disk array controllers.
rst0	Name of cartridge tape. The letters **ct** specify that the data is to be written to the cartridge tape rst0.
sa	Sector address. This parameter must be specified in the case of transfers to a system disk; it specifies the starting logical sector address of the data. This parameter is ignored for tape sources; the write starts at the current tape position.
cma	Central memory address. This parameter specifies the starting central memory word address from which the data is to be read. The default memory word is the current base value. (See the **base** command.)
count	Word count. This parameter specifies the number of 64-bit words to transfer from central memory. This parameter must be specified.

EXAMPLES

The following command line transfers decimal 1.3 million words of data from central memory word address (hexadecimal) 100 to the system disk on controller 2, device 1, for the sector address (hexadecimal) 53BE.

```
    sm p21,53be,100,  .1300000
```

The following command transfers data from central memory word address (hexadecimal) F00 to the cartridge tape. The write starts at the current tape position and an EOF follows the execution of the command.

```
    sm rst0,, f00
```

NOTES

This command accesses central memory through the data channel; therefore, the CPU clock must be ON.

The sector address specified in these commands assumes a sector length of 4096 bytes. The system console must compute the correct physical sector address based on the disk type.

For the commands which write data to the disk, if the count parameter is not a multiple of 512 64-bit words, the data written into the rest of the last sector will be unpredictable.

ERROR MESSAGES

```
The cpu clock is off

Illegal Unit Descriptor

Illegal Controller Number

Illegal Unit number

Error reading memory at address aaaaaaaa

Error writing memory at address aaaaaaaa

Error reading disk at sector ssssssss

Error writing disk at sector ssssssss
```

SEE ALSO

base()

NAME

smt – Saves memory text

SYNOPSIS

smt *filespec*, [*start*], [*count*], [**c**]

DESCRIPTION

The **smt** command saves memory contents to a file.

filespec	Name of the file to which the memory contents are written. If no extension is specified with file name, **.xxx** is appended.
start	Hexadecimal or octal start address to save memory contents. (Default: 0.)
count	Number of words in decimal to store. (Default: 16.)
c	Writes the check bits into the file. (Default: check bits are not written into the file.)

NOTES

This command accesses central memory through the data channels; therefore, the CPU clock must be on.

Memory contents are saved in the radices set by the **mode** command. Use the **mode** command to change the radices.

SEE ALSO

mode()

NAME

smts – Saves memory text using scan mode

SYNOPSIS

smts *filespec*, [*start*], [*count*], [**c**]

DESCRIPTION

The **smts** command saves memory contents to a file using scan move.

filespec	Name of the file to which the memory contents are written. If no extension is specified with file name, **.xxx** is appended.
start	Hexadecimal or octal start address to save memory contents. (Default: 0.)
count	Number of words in decimal to store. (Default: 16.)
c	Writes the check bits into the file. (Default: check bits are not written into the file.)

NOTES

This command is at the scan level; therefore, the CPU clock must be off.

Memory contents are saved in the radices set by the **mode** command. Use the **mode** command to change the radices.

SEE ALSO

mode()

NAME

stat – Displays the CPU and program states

SYNOPSIS

stat [**cpu** | **disk**]

DESCRIPTION

The **stat** command reads and displays CPU every half second continuously until a **<CONTROL> C** is received.

cpu Displays CPU status (**cpu** is default parameter).

disk Displays disk configuration in the system

EXAMPLES

```
                              CPU Status

  Machine States             Breakpoint Information          Program States

  SYSRST           0         ENDP           0                P        069614

  IORST            0         ENDOPNDR       0                NIP      1

  CKRUN            1         ENOPNDW        0                IBA      000000

  CPUSTOPD         0         ENIOR          0                DBA      000000

  INSTSTEP         0         ENIOW          0                XA       13

  SERIAL           0         IUSTOP         0                MM       0

  SCANNING         0         CLKSTOP        0                IFP      0

  RDINGLBA         0         PMATCHED       0                IOR      1

  CKMAR<7:6>       3         OPNDRMATCHD    0                BDM      0

        <5:0>      00        OPNDWMATCHD    0                FPS0

  CKMASK<23:16>    FF        IORMATCHD      0

        <15:8>     FF        IOWMATCHD      0

        <7:0>      FF
```

```
                                   Disk Status
CTLR/DEV            (disk status after initialization)

0/0=00  0/1=00

1/0=FF  1/1=FF

2/0=00  2/1=00  2/2=00  2/3=00

3/0=FF  3/1=FF  3/2=FF  3/3=FF

4/0=FF  4/1=FF  4/2=FF  4/3=FF

5/0=FF  5/1=FF  5/2=FF  5/3=FF

6/0=FF  6/1=FF  6/2=FF  6/3=FF

7/0=FF  7/1=FF  7/2=FF  7/3=FF

B/0=FF  B/1=FF  B/2=FF  B/3=FF

C/0=FF  C/1=FF  C/2=FF  C/3=FF

D/0=FF  D/1=FF  D/2=FF  D/3=FF

E/0=FF  E/1=FF  E/2=FF  E/3=FF

F/0=FF  F/1=FF  F/2=FF  F/3=FF
```

NAME

std – Reads the time and date from the IOS, and writes to central memory

SYNOPSIS

std *addr*

DESCRIPTION

The **std** command reads the time and date from the real time clock in the I/O Subsystem and writes the information to a location in central memory in Wyman clock format.

addr	The central memory address.

EXAMPLES

The following command line writes the time and date to the hex location A4B5DF:

```
std A4B5DF
```

The following command line writes the time and date to octal location 76543210:

```
std o76543210
```

NOTES

The **std** command accesses central memory through the data channels; therefore, the CPU clock must be on.

NAME

step – Executes one or more instructions, one at a time

SYNOPSIS

step [*n*]

DESCRIPTION

The **step** command single steps the CPU through instructions the specified number of times.

n	Number of instructions to execute. (Default is 1.)

NOTES

Clock must be on and IU must be off.

NAME

sync – Flushes all outstanding I/O to hard disk

SYNOPSIS

sync

DESCRIPTION

The **sync** command flushes only local IOS buffers to the SCSI disk. It does not affect UNICOS or system disks.

NAME

tar – Archives tape files

SYNOPSIS

tar [*key*] [*files*]

DESCRIPTION

The **tar** command saves and restores files on magnetic tape and disk files. Its actions are controlled by the *key* argument. The *key* is a string of characters containing one function letter (**c, t,** or **x**) and possibly followed by one or more function modifiers (**v, f, b**). Other arguments to the command are *files* (or directories) specifying which files are to be dumped or restored. In all cases, appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

The function portion of the *key* is specified by one of the following letters:

x Extract. The named *files* are extracted from the archive. If a named file matches a directory whose contents had been written onto the archive, this directory is (recursively) extracted. Use the file or directory's relative path when appropriate, or **tar** will not find a match. The owner, modification time, and mode are restored (if possible). If no *files* argument is specified, the entire content of the archive is extracted. If several files with the same name are on the archive, the last one overwrites all earlier ones.

t Table. The names and other information for the specified files are listed each time that they occur on the archive. The listing is similar to the format produced by the **ls –l** command. If no *files* argument is specified, all the names on the archive are listed.

c Create a new archive; writing begins at the beginning of the archive, instead of after the last file.

The following characters may be used in addition to the letter that selects the desired function:

v Verbose. Normally, **tar** does its work silently. The v option causes it to display the name of each file it treats, preceded by the function letter. With the **t** function, **v** gives more information about the tape entries than just the name.

f File. This causes **tar** to use the *device* argument as the name of the archive.

b Blocking factor. This causes **tar** to use the *block* argument as the blocking factor for tape records. The default and maximum value is 20. The block size is determined automatically when reading tapes created on block special devices (key letters **x** and **t**).

DIAGNOSTICS

Complains about bad key characters and tape read/write errors.

BUGS

There is no way to request the *n*th occurrence of a file.

The length of a file name is currently limited to 8 characters.

tar does not copy empty directories or special files.

EXAMPLES

To extract files from the cartridge tape, you would enter the following:

```
tar -vf rst0
```

To extract only the file **td.c** from a cartridge tape, you would enter the following:

```
tar -xvf rst0 td.c
```

NAME

test – Returns value of program counter or state of flag

SYNOPSIS

test p|pm

DESCRIPTION

Returns the value of the CRAY XMS program counter or the state of the pmatched flag in the CRAY XMS system.

p	Specifies the program counter
pm	Specifies the pmatched flag

EXAMPLES

The following command line returns the value of the program counter:

```
test p
```

The returned value can then be used in an **if** statement following the **test** statement in a command file, as in the following:

```
test pm

if 0 goto :notmatched

echo matched

:notmatched

echo notmatched
```

NOTES

The **test** command only executes in a command file.

For the pmatched flag, 1 = matched and 0 = no match.

NAME

time – Displays or sets the real time clock

SYNOPSIS

time [*dd*/*mm*/*yy*], [*hh*:*mm*:*ss*]

DESCRIPTION

The **time** command displays or sets the real time clock in the I/O Subsystem. If no parameters are used, the system date and time are returned.

dd/*mm*/*yy*	Day, month, and year
hh:*mm*:*ss*	Hours, minutes, and seconds

NOTES

Note that the separator is a slash for day, month, and year; and a colon for the hours, minutes, and seconds. Two digits must be used in all fields.

NAME

umount – Unmounts local Winchester drive

SYNOPSIS

umount c:

DESCRIPTION

The **umount** command flushes buffered I/O out to the Winchester drive and then unmounts it.

EXAMPLES

```
umount c:
```

NAME

ver – Displays version number of the IOS

SYNOPSIS

ver

DESCRIPTION

The **ver** command displays the version level of the IOS you are currently running along with the date stamp it was built.

NAME

wait – Causes command processing to wait

SYNOPSIS

wait [*seconds*]

DESCRIPTION

The **wait** command causes command processing to wait a specified number of seconds before executing the next command.

seconds	Specifies the number of seconds to wait. (Default: 10).

EXAMPLES

The following command line causes a wait for 15 seconds before the next command in the command file is executed.

```
wait 15
```

Reader's Comment Form

CRAY XMS Systems IOS Reference Manual SR–3085 1.1

Your reactions to this manual will help us provide you with better documentation. Please take a moment to complete the following items, and use the blank space for additional comments.

List the operating systems and programming languages you have used and the years of experience with each.

Your experience with Cray Research computer systems: _____0-1 year _____1-5 year _____5+years

How did you use this manual: _____in a class _____as a tutorial or introduction _____as a procedural guide _____as a reference _____for troubleshooting _____other

Please rate this manual on the following criteria:

	Excellent			Poor
Accuracy	4	3	2	1
Appropriateness (correct technical level)	4	3	2	1
Accessibility (ease of finding information)	4	3	2	1
Physical qualities (binding, printing, illustrations)	4	3	2	1
Terminology (correct, consistent, and clear)	4	3	2	1
Number of examples	4	3	2	1
Quality of examples	4	3	2	1
Index	4	3	2	1

Please use the space below for your comments about this manual. Please include general comments about the usefulness of this manual. If you have discovered inaccuracies or omissions, please specify the number of the page on which the problem occurred.

Name ______________	Address ______________
Title ______________	City ______________
Company ______________	State/Country ______________
Telephone ______________	Zip code ______________
Today's date ______________	Electronic mail address ______________

Cut along this line

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 6184 ST. PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY
RESEARCH, INC.

ATTN: Software Information Services
655 LONE OAK DR BLDG F
EAGAN MN 55121-9957

Fold

Reader's Comment Form

CRAY XMS Systems IOS Reference Manual SR–3085 1.1

Your reactions to this manual will help us provide you with better documentation. Please take a moment to complete the following items, and use the blank space for additional comments.

List the operating systems and programming languages you have used and the years of experience with each.

Your experience with Cray Research computer systems: _____0-1 year _____1-5 year _____5+years

How did you use this manual: _____in a class _____as a tutorial or introduction _____as a procedural guide _____as a reference _____for troubleshooting _____other

Please rate this manual on the following criteria:

	Excellent			Poor
Accuracy	4	3	2	1
Appropriateness (correct technical level)	4	3	2	1
Accessibility (ease of finding information)	4	3	2	1
Physical qualities (binding, printing, illustrations)	4	3	2	1
Terminology (correct, consistent, and clear)	4	3	2	1
Number of examples	4	3	2	1
Quality of examples	4	3	2	1
Index	4	3	2	1

Please use the space below for your comments about this manual. Please include general comments about the usefulness of this manual. If you have discovered inaccuracies or omissions, please specify the number of the page on which the problem occurred.

Name ______________________	Address ______________________
Title ______________________	City ______________________
Company ______________________	State/Country ______________________
Telephone ______________________	Zip code ______________________
Today's date ______________________	Electronic mail address ______________________

Cut along this line

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 6184 ST. PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

ATTN: Software Information Services
655 LONE OAK DR BLDG F
EAGAN MN 55121-9957

Fold